A Million Ways to Provision Embedded Linux Devices

Drew Moseley
Technical Solutions Architect
Mender.io

Session Overview

- 1,000,000 barely feels like an exaggeration
- Anatomy of an embedded Linux system
- Discuss storage and provisioning of:
  - Bootloader
  - Kernel/DTB
  - Root filesystem
- Describe specific boards using different models

Goal: Present an overview of all the provisioning models you will encounter

About Me

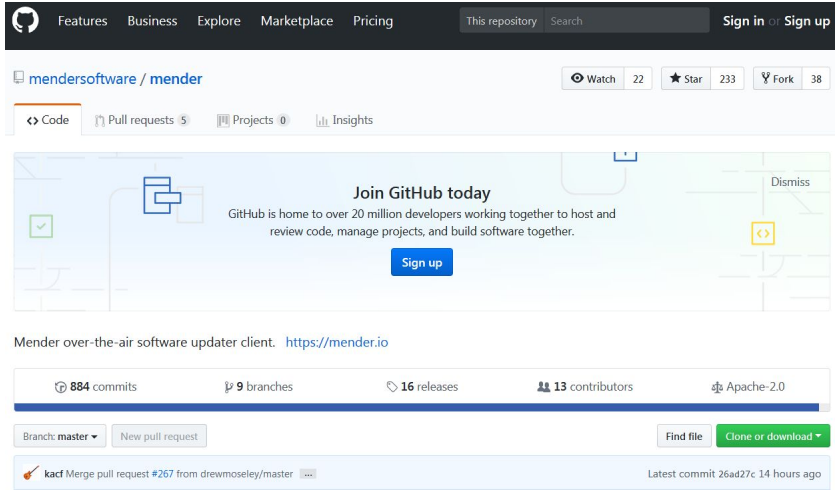
Drew Moseley

- 10 years in Embedded Linux/Yocto development.
- Longer than that in general Embedded Software.
- Project Lead and Solutions Architect.

drew.moseley@mender.io
https://twitter.com/drewmoseley
https://www.linkedin.com/in/drewmoseley/
https://twitter.com/mender_io

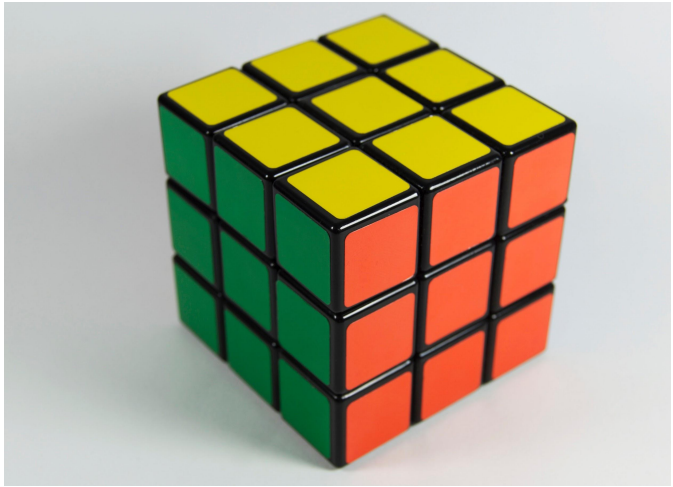
Mender.io

- Over-the-air update manager for embedded Linux
- Open source (Apache 2.0 License)
- Remote deployment management (server)
- Under active development



Challenges for Embedded Linux Developers

- No “one way” to get initial images onto boards.
- Mechanisms may vary between development, manufacturing and CI/QA.
- Mechanisms vary widely between
  - Boards
  - Manufacturers
- Slow provisioning -> long development cycles



Anatomy of a System

Root filesystem: all files, executables, data, etc for the system

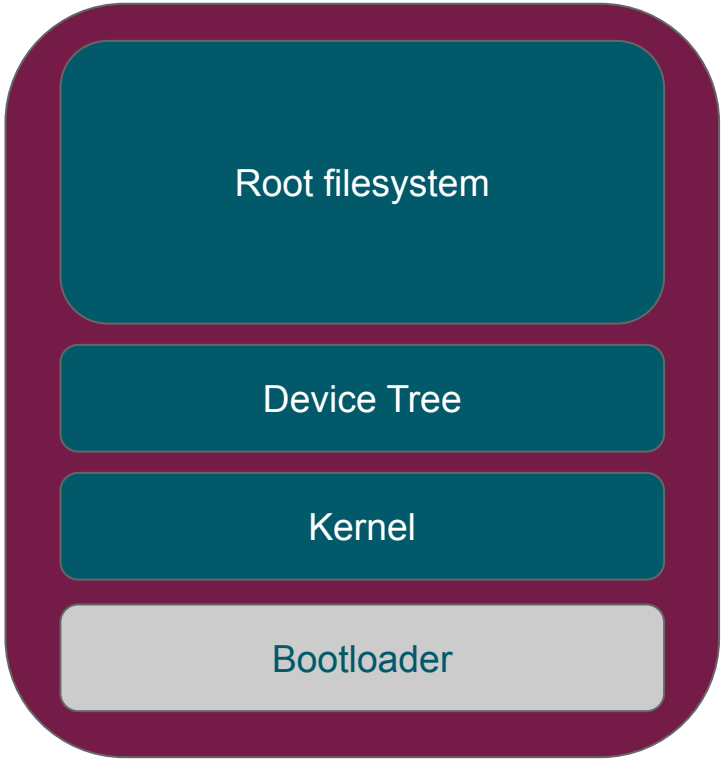
Device Tree: hardware description

Kernel: core operating system functionality

- Resource management
- Process control
- Device drivers

Bootloader: system initialization code starting with the reset vector.

- Initialize and scrub RAM
- Setup power rails and clocks
- Load the “rest” of the system



Storage Overview

Bootloader:

1. Separate flash/interface. ie SPI Boot Flash - Compulab CL-SOM-iMX6
2. MTD partition - Compulab CL-SOM-iMX7
3. Inter-partition space - Toradex Colibri i.MX6/eMMC
4. UBI partition - Toradex Colibri i.MX7/NAND
5. File in a partition - Beaglebone

Kernel/DTB:

1. Separate partition - Compulab, Toradex
2. Files in a separate partition - Beaglebone
3. Files in the root filesystem - Mender-enabled boards
4. Files retrieved over network - most boards

Root filesystem:

1. eMMC/SDCard - Beaglebone, Raspberrypi, etc
2. Raw NAND flash - Toradex Colibri
3. USB drive - Raspberrypi (not the default)
4. SATA/SSD - NUC, PC
5. NFS - most boards

Yocto + QEMU - setup

Simple setup for development (9 minute build time with SSTATE):

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
     build-essential chrpath socat cpio python python3 python3-pip python3-pexpect \
     xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev \
     xterm
git clone git://git.yoctoproject.org/poky -b yocto-2.7.1
source poky/oe-init-build-env
cat >> conf/local.conf <<'EOF'
SSTATE_MIRRORS = "\
     file://.* http://sstate.yoctoproject.org/2.6/PATH;downloadfilename=PATH \n \
     "
EOF
MACHINE=qemux86 bitbake core-image-base
```

https://www.yoctoproject.org/docs/2.7.1/brief-yoctoprojectqs/brief-yoctoprojectqs.html
http://bit.ly/yocto-qemu-deploy

Yocto + QEMU - run

Standard Block Device:

```
$ runqemu qemux86 nographic
```

NFS mounted root filesystem[1]:

```
$ sudo apt-get install rpcbind
$ echo 'OPTIONS="-i -w"' | sudo tee -a /etc/default/rpcbind
$ sudo service portmap restart
$ runqemu qemux86 nfs nographic
```

The runqemu script hides all the complication of setting this up.

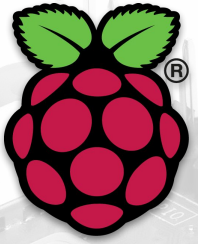
Differences from typical board scenario:

1. No bootloader
2. No DTB
3. Kernel loaded directly from host filesystem

Excellent method for development of non-HW-specific bits.

[1]Note that this has worked for me in the past but in my current setup, the system fails to boot.
The above was run on Ubuntu 18.04

Provisioning Model: SD Card



Platform used: Raspberry Pi 3

Proprietary bootloader in ROM

- Loads kernel or u-boot binary as a file from FAT
- DTB loaded as file from FAT

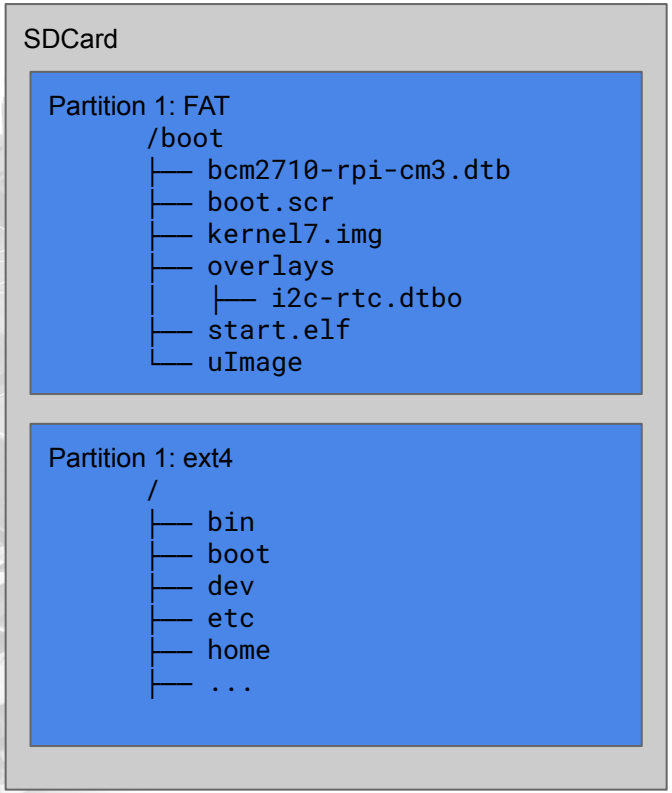
Root filesystem mounted from SD/MMC

Image file generated directly by Yocto

Provisioning done on build platform:

```
$ sudo dd if=<file>.rpi-sdimg of=/dev/sdb bs=8M
```

SDCard inserted in board and booted as-is



Provisioning Model: eMMC

Platform used: TechNexion PICO-PI-IMX7

Bootloader in unpartitioned space on eMMC

Root filesystem mounted from eMMC

Image file generated directly by Yocto

Load U-Boot image from build platform into RAM:

```
$ sudo imx_usb SPL
$ sudo imx_usb u-boot.img
```

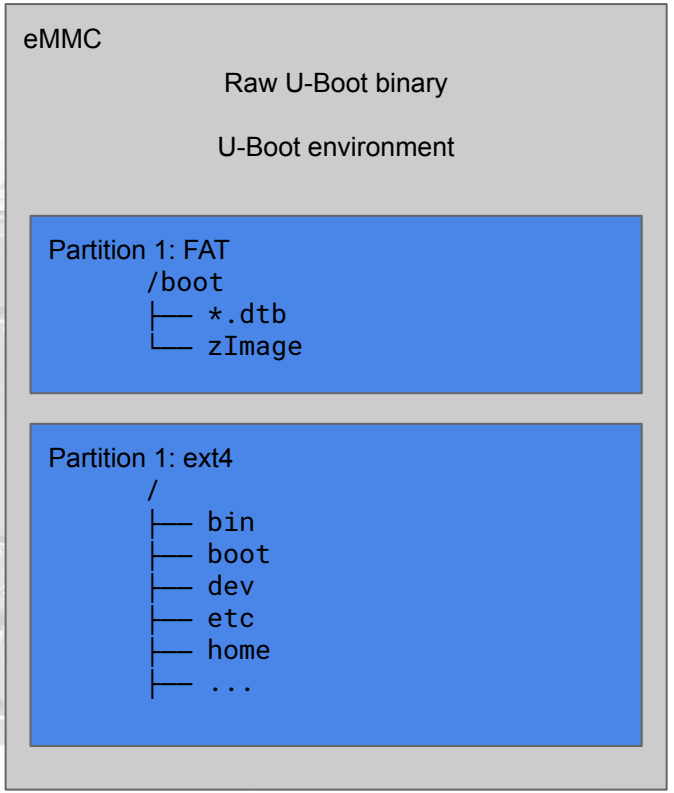
From RAM-based U-Boot, install new images

Target board configured as USB gadget mode

```
usb start
ums 0 mmc 0
```

Provisioning done on build platform:

```
$ sudo dd if=<file>.rpi-sdimg of=/dev/sdb bs=8M
```



Provisioning Model: SPI + eMMC

Platform used: Compulab IOT-GATE-iMX7

Jumper selection for bootloader location: SPI flash or SD

Bootloader configuration determines eMMC or SD for:

- Kernel
- DTB
- Root filesystem

Boot full SD-based image provided by Compulab

Copy Yocto-generated images to USB key

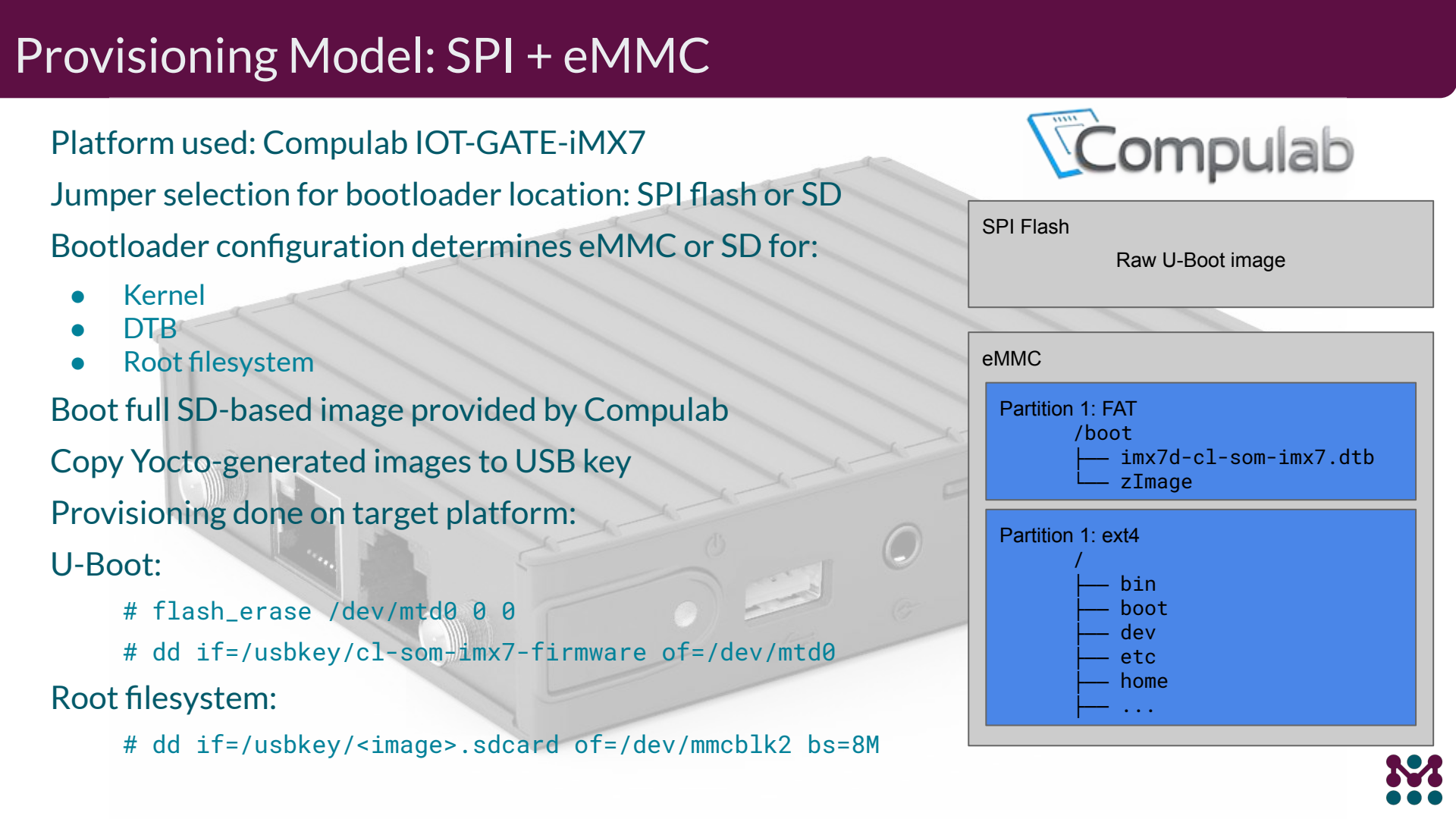
Provisioning done on target platform:

U-Boot:

```
# flash_erase /dev/mtd0 0 0
# dd if=/usbkey/cl-som-imx7-firmware of=/dev/mtd0
```

Root filesystem:

```
# dd if=/usbkey/<image>.sdcard of=/dev/mmcblk2 bs=8M
```



Provisioning Model: Raw NAND

Platform used: Toradex Colibri iMX7 + Aster baseboard

Copy Yocto-generated images to SD Card

Short test points to boot in recovery mode

Load U-Boot image from build platform into RAM:

```
$ sudo imx_usb u-boot-nand.imx
```

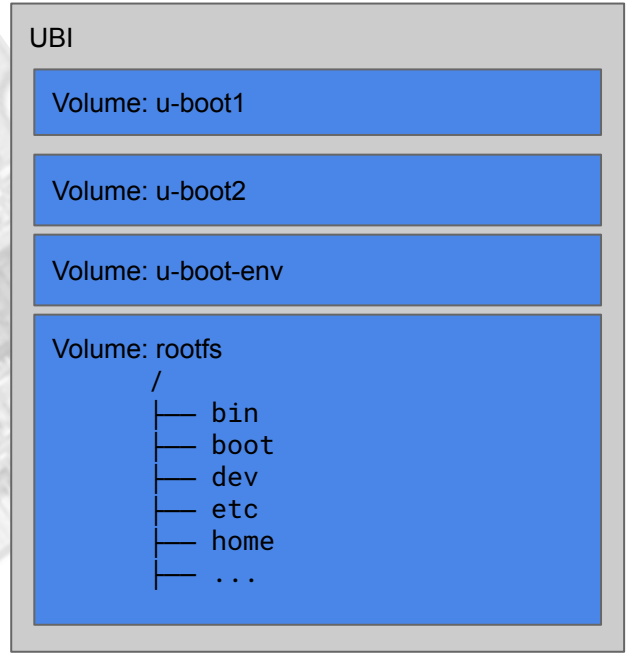
From RAM-based U-Boot, install new images

U-Boot:

```
nand erase.part u-boot1; nand erase.part u-boot2
load mmc 0:1 ${loadaddr} u-boot-nand.imx
nand write ${loadaddr} u-boot1 ${filesize}
nand write ${loadaddr} u-boot2 ${filesize}
nand erase.part u-boot-env
```

Root filesystem:

```
nand erase.part ubi
load mmc 0:1 ${loadaddr} <image>.ubi
nand write ${loadaddr} ubi ${filesize}
```



Provisioning Model: Android Tools

Platform used: Dragonboard 410c[1]

Switch on board to boot in fastboot mode

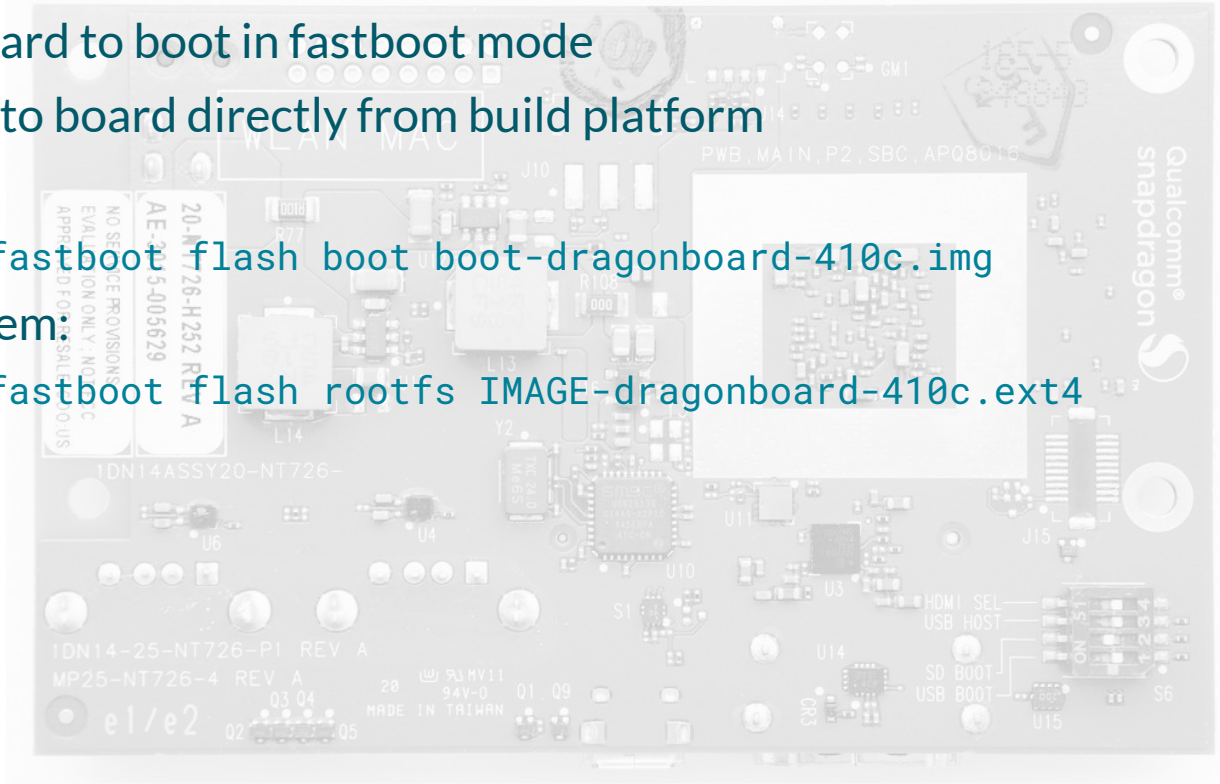
Load images to board directly from build platform

U-Boot:

```
$ sudo fastboot flash boot boot-dragonboard-410c.img
```

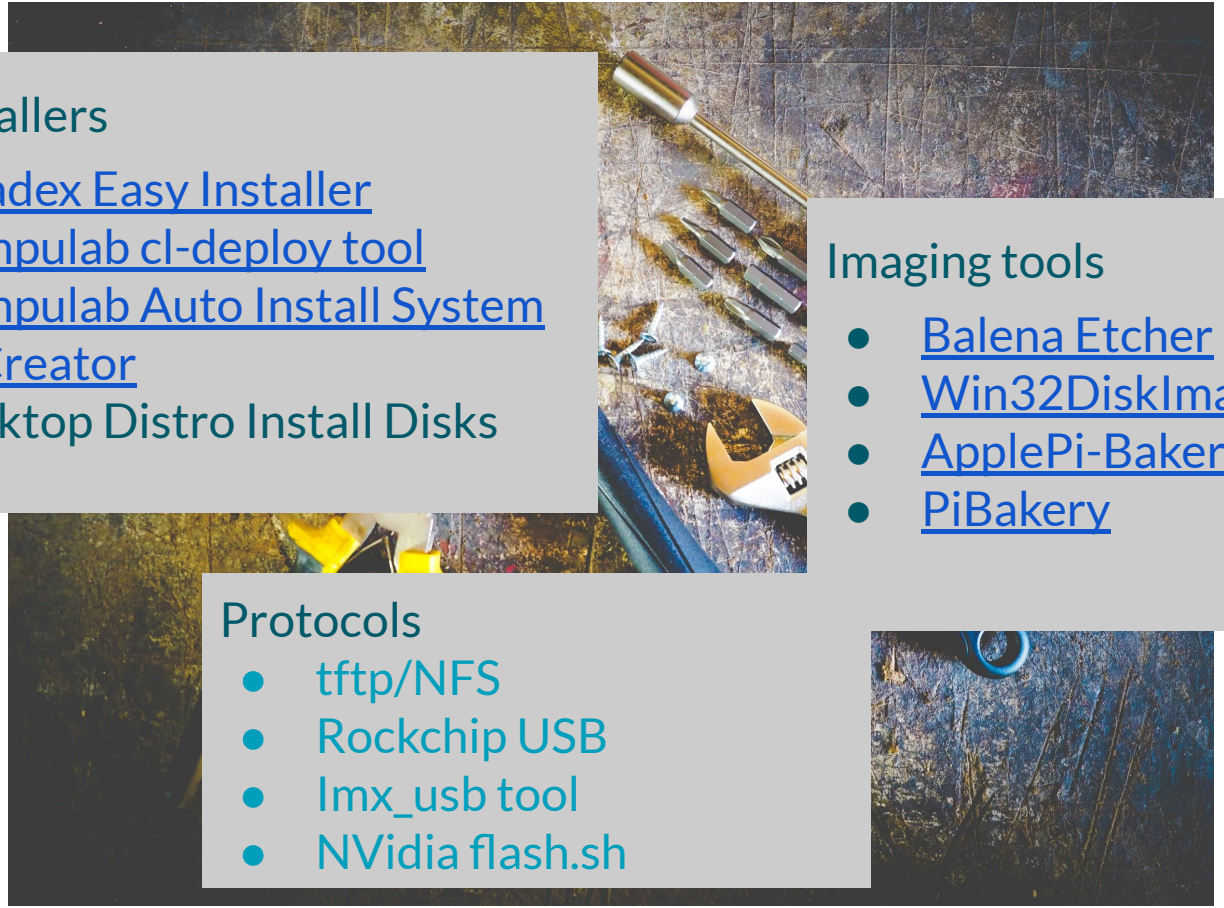
Root filesystem:

```
$ sudo fastboot flash rootfs IMAGE-dragonboard-410c.ext4
```



[1] https://www.96boards.org/documentation/consumer/dragonboard/dragonboard410c/build/open-embedded.md.html

Misc Provisioning Tools



Live Installers

- Toradex Easy Installer
- Compulab cl-deploy tool
- Compulab Auto Install System
- QtCreator
- Desktop Distro Install Disks

Imaging tools

- Balena Etcher
- Win32DiskImager
- ApplePi-Baker
- PiBakery

Protocols

- tftp/NFS
- Rockchip USB
- Imx_usb tool
- NVidia flash.sh

Other Considerations

Product Development:

- CI/CD integration
- Systems developers vs Application developers
- Heterogenous targets

Manufacturing:

- Unattended installation
- Per-board data
- Registration with infrastructure
- Burn-in test vs production images

Thank You!

Q&A

@drewmoseley

drew.moseley@mender.io

@mender_io

https://mender.io
https://docs.mender.io/2.0/getting-started
https://hub.mender.io